IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re U.S. Patent Application of	)
	)
MASUDA et al.	)
	)
Application Number: To be Assigned	)
	)
Filed: Concurrently Herewith	)
	)
For: LOAD DISTRIBUTION METHOD AND CLIENT-SERVER SYSTEM	)
	)
ATTORNEY DOCKET NO. NITT.0195	)

Honorable Assistant Commissioner
for Patents
Washington, D.C. 20231

REQUEST FOR PRIORITY
UNDER 35 U.S.C. § 119
AND THE INTERNATIONAL CONVENTION

Sir:

In the matter of the above-captioned application for a United States patent, notice is hereby given that the Applicant claims the priority date of August 11, 2003, the filing date of the corresponding Japanese patent application 2003-291286.

A certified copy of Japanese patent application 2003-291286 is being submitted herewith. Acknowledgment of receipt of the certified copy is respectfully requested in due course.

Respectfully submitted,

Stanley P. Fisher
Registration Number 24,344

Juan Carlos A. Marquez
Registration Number 34,072

REED SMITH LLP
3110 Fairview Park Drive
Suite 1400
Falls Church, Virginia 22042
(703) 641-4200
February 23, 2004

PATENT OFFICE
JAPANESE GOVERNMENT

This is to certify that the annexed is a true copy of the following application as filed with this office.

Date of Application : August 11, 2003
Application Number : Patent Application No. 2003-291286
Applicant (s) : Hitachi, Ltd.

Dated this 13th day of February, 2004

Yasuo IMAI
Commissioner,
Patent Office

Certificate No. 2004-3009134

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日 Date of Application:	２００３年　８月１１日
出願番号 Application Number:	特願２００３－２９１２８６
[ST. 10/C]:	[ＪＰ２００３－２９１２８６]
出願人 Applicant(s):	株式会社日立製作所

２００４年　２月１３日

特許庁長官
Commissioner,
Japan Patent Office　今井康夫

出証番号　出証特２００４－３００９１３４

【書類名】　特許願
【整理番号】　GM0306054
【提出日】　平成15年 8月11日
【あて先】　特許庁長官殿
【国際特許分類】　G16F　9/46
【発明者】
　【住所又は居所】　東京都国分寺市東恋ヶ窪一丁目２８０番地　株式会社日立製作所中央研究所内
　【氏名】　増田　峰義
【発明者】
　【住所又は居所】　東京都国分寺市東恋ヶ窪一丁目２８０番地　株式会社日立製作所中央研究所内
　【氏名】　垂井　俊明
【発明者】
　【住所又は居所】　東京都国分寺市東恋ヶ窪一丁目２８０番地　株式会社日立製作所中央研究所内
　【氏名】　樋口　達雄
【特許出願人】
　【識別番号】　000005108
　【氏名又は名称】　株式会社日立製作所
【代理人】
　【識別番号】　100075513
　【弁理士】
　【氏名又は名称】　後藤　政喜
【選任した代理人】
　【識別番号】　100084537
　【弁理士】
　【氏名又は名称】　松田　嘉夫
【選任した代理人】
　【識別番号】　100114236
　【弁理士】
　【氏名又は名称】　藤井　正弘
【手数料の表示】
　【予納台帳番号】　019839
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　特許請求の範囲　1
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
　【包括委任状番号】　0110326

【書類名】特許請求の範囲
【請求項１】
複数のクライアントと、
前記クライアントからのリクエストを処理する複数のサーバを含み、前記複数のサーバの数を動的に変更するサーバ・クラスタと、によって構成されるクライアント・サーバシステムに用いられる負荷分散方法であって、
前記クライアントは、
前記サーバ・クラスタを構成するサーバの数を検出し、
サーバ数の増加が検出された直後は、該増加したサーバに対して送出されるリクエストの配分を他のサーバに比べて小さく設定し、
前記設定された配分に基づいて前記複数のサーバに対してリクエストを送出することを特徴とする負荷分散方法。
【請求項２】
前記クライアントは、前記増加したサーバに対して送出されるリクエストの配分を、時間の経過と共に増加するように設定することを特徴とする請求項１に記載の負荷分散方法。
【請求項３】
前記クライアントは、前記サーバ・クラスタのサーバの数の増加が検出されたことを契機として、該増加したサーバに対して送出されるリクエストの配分を、他のサーバに比べて小さく設定することを特徴とする請求項１又は２に記載の負荷分散方法。
【請求項４】
前記クライアントは、
前記増加したサーバの性能に関する情報を取得し、
該取得した情報に基づいて、該増加したサーバに対して送出されるリクエストの配分を設定することを特徴とする請求項１から３のいずれか一つに記載の負荷分散方法。
【請求項５】
前記クライアントは、
前記増加したサーバの状態に関する情報を取得し、
該取得した情報に基づいて、該増加したサーバに対して送出されるリクエストの配分を設定することを特徴とする請求項１から３のいずれか一つに記載の負荷分散方法。
【請求項６】
前記クライアントは、前記サーバの状態に関する情報として、キャッシュヒット率、キャッシュ使用率又はリクエストの待ち数に関する情報の一つ以上を取得することを特徴とする請求項５に記載の負荷分散方法。
【請求項７】
前記クライアント・サーバシステムは、前記サーバの数を管理する管理サーバを備え、
前記クライアントは、前記管理サーバから、前記サーバ・クラスタのサーバの数の増加の通知を受信したことを契機として、該増加したサーバに対して送出されるリクエストの配分を、他のサーバに比べて小さく設定することを特徴とする請求項１又は２に記載の負荷分散方法。
【請求項８】
前記クライアント・サーバシステムは、前記サーバの性能に関する情報を取得する管理サーバを備え、
前記クライアントは、
前記管理サーバから、前記増加したサーバの性能に関する情報を取得し、
該取得した情報に基づいて、該増加したサーバに対して送出されるリクエストの配分を設定することを特徴とする請求項１又は２に記載の負荷分散方法。
【請求項９】
前記クライアントは、前記サーバとの間の通信接続数を設定することによって、前記増加したサーバに対して送出されるリクエストの配分を設定することを特徴とする請求項１

から８のいずれか一つに記載の負荷分散方法。
【請求項１０】
前記クライアントは、前記サーバに送出されるリクエストの各サーバに対する割当を変更することによって、前記各サーバに対して送出されるリクエストの配分を設定することを特徴とする請求項１から８のいずれか一つに記載の負荷分散方法。
【請求項１１】
前記クライアント・サーバシステムは、前記サーバに接続されるストレージ装置を備え、
前記サーバは、前記ストレージ装置に記憶されるファイルの格納場所を示すディレクトリ情報を保持し、
前記クライアントは、前記サーバに送出されるリクエストの各サーバに対する割当として、各サーバへの前記ディレクトリ情報を格納する割当を変更することによって、前記各サーバに対して送出されるリクエストの配分を設定することを特徴とする請求項１０に記載の負荷分散方法。
【請求項１２】
複数のクライアントと、
前記クライアントからのリクエストを処理する複数のサーバを含み、前記複数のサーバの数を動的に変更するサーバ・クラスタと、によって構成されるクライアント・サーバシステムであって、
前記クライアントは、
前記各サーバに対して送出されるリクエストの配分を設定する負荷設定部と、
前記サーバ・クラスタを構成するサーバの数を検出する台数検出部と、
前記負荷設定部によって設定された配分に基づいて、前記複数のサーバに対してリクエストを送出する負荷分散部と、を備え、
前記負荷設定部は、前記台数検出部によってサーバ数の増加が検出された直後は、該増加したサーバに対して送出されるリクエストの配分を他のサーバに比べて小さく設定することを特徴とするクライアント・サーバシステム。
【請求項１３】
前記クライアントは、前記サーバに送出されるリクエストの各サーバに対する割当てを保持する割当保持部を備え、
前記負荷分散部は、前記リクエストの前記サーバに対する割当を変更することによって、前記各サーバに対して送出するリクエストの配分を設定することを特徴とする請求項１２に記載のクライアント・サーバシステム。
【請求項１４】
前記クライアント・サーバシステムは、前記サーバに接続されるストレージ装置を備え、
前記サーバは、前記ストレージ装置に記憶されるファイルの格納場所を示すディレクトリ情報を保持するディレクトリ情報保持部を備え、
前記クライアントは、前記サーバに送出されるリクエストの各サーバに対する割当として、前記ディレクトリ情報を格納しているサーバの割当を保持する割当管理部を備え、
前記負荷分散部は、前記ディレクトリ情報を格納しているサーバの割当を変更することによって、前記各サーバに対して送出するリクエストの配分を設定することを特徴とする請求項１３に記載のクライアント・サーバシステム。

【書類名】明細書
【発明の名称】負荷分散方法及びクライアント・サーバシステム
【技術分野】
【０００１】
本発明は、インターネット上での電子商取引等のサービス構築に用いられる、サービス利用者からのリクエストを処理するサーバ・クラスタシステム等を用いたクライアント・サーバシステムにおいて、サービス需要の増減に応じてサーバ・クラスタシステムを構成するサーバの台数を変更するクラスタ再構成技術に適合した、クライアントとサーバ・クラスタシステム間の負荷分散方法及び負荷分散方法を実装したクライアント・サーバシステム関する。
【背景技術】
【０００２】
複数のサーバをネットワークで結合し、見かけ上単一のサーバを構成するサーバ・クラスタシステムが、インターネット上で電子商取引等の各種サービスを提供する計算機システムに用いられている。計算機システムとしてサーバ・クラスタシステムを用いる場合、クライアントとサーバ・クラスタシステムとの間で負荷分散を行うことが一般的である。具体的には、クライアントからのサービス処理要求（リクエスト）を、サーバ・クラスタシステムを構成する複数のサーバの各々の処理能力に応じて振り分けるよう、リクエストの配分を決定する。
【０００３】
この負荷分散に用いるアルゴリズム、すなわち、どのサーバにどれだけのリクエストを送出するかを決定する手順は、システムの性能を左右する。負荷分散アルゴリズムが適切でない場合、クライアントからのリクエストが均等にサーバに振り分けられず、サーバ間で負荷の不均衡が生じる。負荷の重いサーバでは、負荷が軽い他のサーバと比較すると、リクエストに対する処理時間が大幅に大きくなり、リクエスト処理が遅れてしまう。例えば、インターネット上でのサービスの場合、リクエスト処理の遅れは、サービス利用者への応答の遅れとして顕在化する。
【０００４】
この、システムにとって重要な要素である負荷分散アルゴリズムの代表的なものにラウンドロビン方式が知られている。ラウンドロビン方式とは、同時に入力する複数の要求信号の内の一つを予め設定された優先順位に従って、かつ均等に選択されるように、優先順位の並び替えを行う方式である。このラウンドロビン方式を利用すると、負荷分散先のサーバに、リクエストを順番に送出することができる（例えば、特許文献１参照。）。
【０００５】
また、ラウンドロビン方式を応用した方式に、重み付きラウンドロビン方式がある。重み付きラウンドロビン方式とは、ラウンドロビン方式の優先順位を、所定の重みに従って並び替えを行う方式である。この重み付けラウンドロビン方式を利用すると、サーバのプロセッサの動作周波数やメモリ量などを基に、サーバの性能値を“重み”として算出しておき、サーバに、重みに応じた数のリクエストを振り分けることができる（例えば、特許文献２参照。）。
【０００６】
また、近年、クラスタ再構成技術と呼ばれる、サーバ・クラスタシステムを動的に再構成する技術が提案されている。このクラスタ再構成技術は、クライアントからのサービス要求を、負荷分散装置によってサーバ・クラスタで稼働中のいずれかのサーバに振り分け、アクセスの負荷に応じてサーバ・クラスタの構成（台数）を変更する技術である。このクラスタ再構成技術は、インターネット上でのサービスを提供するシステムのように、サービス利用者数の変動が大きいシステムでは特に有効である（例えば、特許文献３参照。）。
【０００７】
また、ＮＡＳ（Network Attached Storage）にサーバ・クラスタシステムを適用し、容

量拡張、減縮が容易に行えるよう、ファイルの識別子にハッシュ関数を適応し、ファイルと該ファイルの格納先とを組み合わせたテーブルを管理するシステムも知られている（例えば、非特許文献１参照。）。

【特許文献１】特開２０００－２３１４９６号公報
【特許文献２】特開２００１－２１７８３６号公報
【特許文献３】特開２００２－１６３２４１号公報
【非特許文献１】川本　真一，他５名，“ファイル自律配置方式を備えた仮想一元化ＮＡＳシステムＸ－ＮＡＳの実現と評価”，［online］，［平成１５年７月１７日検索］，インターネット＜ＵＲＬ：http://www.ieice.org/iss/de/DEWS/proc/2003/papers/4-B/4-B-01.pdf＞

【発明の開示】
【発明が解決しようとする課題】
【０００８】
前述したクラスタ再構成技術を、不適切な負荷分散アルゴリズムと併用した場合、サーバ・クラスタシステムに新たにサーバを追加した際に、サーバ・クラスタシステム内を構成するサーバ間で負荷の不均衡が発生することがある。このサーバ間での負荷の不均衡は、クラスタ再構成技術を用いてサーバ・クラスタシステムに新たに追加したサーバと、クラスタ内で既に動作していた既存のサーバとの間で、リクエスト処理能力に差があることが原因である。
【０００９】
このような、サーバ間でリクエスト処理能力に差が起こる原因の一つに“キャッシュ”がある。
【００１０】
例として、複数のキャッシュ・サーバで構成されるサーバ・クラスタシステムについて説明する。キャッシュ・サーバは、Ｗｅｂサーバの前段（Ｗｅｂサーバとサービス利用者との間）に配置され、Ｗｅｂサーバの代わりに、サービス利用者からのＷｅｂコンテンツ配信要求に対するＷｅｂコンテンツを配信する。要求されたＷｅｂコンテンツがキャッシュ・サーバ内にキャッシングされている場合、キャッシングされているコンテンツをＷｅｂサーバから取得することなく、サービス利用者に即時に配信できるため、サービス利用者への応答時間は短い。一方、Ｗｅｂコンテンツがキャッシュ・サーバ内にキャッシングされていない場合、Ｗｅｂコンテンツを、一旦Ｗｅｂサーバから取得した後、サービス利用者に配信を行う必要があるため、サービス利用者への応答時間が長くなる。つまり、キャッシュ・サーバ内にＷｅｂコンテンツがキャッシュされているか否かで、サービス利用者への応答時間に大きな差ができる。
【００１１】
このような特性を持つキャッシュ・サーバから構成されるサーバ・クラスタシステムに、新たにキャッシュ・サーバを一台追加した場合、キャッシュ・サーバを追加した直後は、追加されたキャッシュ・サーバ内にはＷｅｂコンテンツは全くキャッシングされていない。そのため、追加直後のキャッシュ・サーバは、利用者からの要求を受けるたびにＷｅｂサーバからコンテンツを取得する必要があり、サービス利用者への応答時間が長くなる。一方、これまでサーバ・クラスタシステム内で稼動していた既存のキャッシュ・サーバは、多くのＷｅｂコンテンツをキャッシングしているので、新たに追加されたキャッシュ・サーバと比較すると、サービス利用者への応答時間が短い。
【００１２】
このように、サービス利用者からの要求を処理する能力に差がある既存キャッシュ・サーバと追加キャッシュ・サーバに、ラウンドロビン方式で同数のリクエストを送出した場合、追加されたキャッシュ・サーバでは、リクエストの処理が間に合わず、サーバ内で未処理リクエストの長い待ち行列が発生する。そのため、待ち行列の理論に従い、サービス利用者への応答時間は指数関数的に長くなる。そのため、応答時間の大幅な遅れが発生し、サービス利用者に多大な不利益を与える、ひいてはサービス提供者の信頼を損なう。

【００１３】
本発明は、このような問題に鑑みてなされたものであり、サーバ・クラスタシステムに新たにサーバが追加された際、追加されたサーバにおいて、リクエスト処理時間が長大になることを回避する分散制御方法を提供する。
【課題を解決するための手段】
【００１４】
本発明は、複数のクライアントと、前記複数のクライアントからのリクエストを処理する複数のサーバを含み、前記複数のサーバの数を動的に変更するサーバ・クラスタと、によって構成されるクライアント・サーバシステムに用いられる負荷分散方法であって、前記クライアントは、前記サーバ・クラスタを構成するサーバの数を検出し、サーバ数の増加が検出された直後は、該増加したサーバに対して送出されるリクエストの配分を他のサーバに比べて小さく設定し、前記設定された配分に基づいて前記複数のサーバに対してリクエストを送出することを特徴とする。
【００１５】
本発明では、サーバ・クラスタシステムにサーバを追加した直後は特別な負荷分散を行う。すなわち、既存のサーバと比較して性能（処理能力）が劣る追加サーバに対するリクエスト送出量を、既存サーバに比べて抑えることで、追加サーバにおいてリクエスト処理時間が長大になることを回避する。
【００１６】
本発明のクライアント・サーバシステムは、クライアントからのリクエストを処理するための複数のサーバから構成されるサーバ・クラスタシステムとクライアントとからなり、サーバ・クラスタシステムを構成するサーバの台数は、リクエスト数の変動に応じて増減する。クライアントには、サーバ・クラスタシステムにリクエストを振り分ける負荷分散機能と、リクエストの振り分け方を制御する負荷制御プログラムとを備える。また、前記負荷制御プログラムはサーバ・クラスタシステムのサーバ台数の変更を検出する機能を持つ。
【００１７】
本発明の特徴は、負荷制御プログラムが追加サーバへのリクエスト送出量を調整し、既存サーバへのリクエスト送出量に比べて、追加直後の追加サーバへのリクエスト送出量を少なくし、時間経過と供に追加サーバへのリクエスト送出量を段階的に増加させることである。
【００１８】
負荷制御プログラムは、サーバ・クラスタシステムを構成するサーバ台数の変更を検知する機能を持つ。例えば、サーバ・クラスタシステムのサーバ台数を管理する管理サーバを備え、管理サーバが、サーバ台数の変更があった場合に、その変更を負荷制御プログラムに通知する。負荷制御プログラムは、サーバ台数変更の通知の検出を契機として、追加サーバへのリクエストの送出を開始する。
【００１９】
負荷制御プログラムは、追加サーバへのリクエスト送出量を段階的に増加させる。この増分の計算方法には二つの方式がある。一つは、追加サーバから性能情報を取得し増分計算に利用する方式、もう一つは、サーバから性能情報を取得しない方式である。
【００２０】
サーバ・クラスタシステムからの性能情報を利用する場合、負荷制御プログラムは、追加サーバから、例えばキャッシュヒット率やリクエストの待ち行列長といった性能情報を取得する機能と、その性能情報を基にリクエスト送出量の増分を計算する機能を持ち、計算結果に従って、追加サーバへのリクエスト送出量を増加させる。
【００２１】
追加サーバからの性能情報を利用しない場合、負荷制御プログラムは、予め定められた規則（例えば、サーバが追加されてから所定の時間内は、リクエストの送出量を１０秒間毎に１０％増加させる）に従い追加サーバへのリクエスト送出量を増加させる。

【発明の効果】
【００２２】
本発明によると、複数のクライアントと、前記複数のクライアントからのリクエストを処理する複数のサーバを含み、前記複数のサーバの数を動的に変更するサーバ・クラスタと、によって構成されるクライアント・サーバシステムに用いられる負荷分散方法であって、前記クライアントは、前記サーバ・クラスタを構成するサーバの数を検出し、サーバ数の増加が検出された直後は、該増加したサーバに対して送出されるリクエストの配分を他のサーバに比べて小さく設定し、前記設定された配分に基づいて前記複数のサーバに対してリクエストを送出することを特徴とするので、サーバ・クラスタシステムに新たにサーバが追加された際に、追加サーバでのリクエスト処理時間が長大となることを回避できる。
【発明を実施するための最良の形態】
【００２３】
本発明の実施の形態について、図面を参照して説明する。
【００２４】
図１は、本発明の第１の実施の形態のクライアント・サーバシステムの概要を表すブロック図である。
【００２５】
本実施の形態のクライアント・サーバシステムは、複数のクライアント１００がサーバ間ネットワーク７００によってサーバ・クラスタシステム１１００と接続されている。
【００２６】
クライアント１００は、サーバ・クラスタシステム１１００にサービスを要求し、サーバ・クラスタシステム１１００から要求に対する結果を受け取る。このクライアント１００では、負荷分散機能３００及び負荷制御プログラム４００が動作している。なお、クライアント１００からサーバ・クラスタシステム１１００に送信するサービスの処理の要求を「リクエストの送出」と呼ぶ。
【００２７】
負荷制御プログラム４００は、サーバ・クラスタシステム１１００を構成する各サーバに対して、各々どれだけの配分でリクエストを送出するのかをクライアント１００が判断するためのデータである、負荷重み表４０５を作成する。また、負荷分散機能３００は、負荷制御プログラム４００が作成した負荷重み表４０５に従って、各々のサーバとクライアントプログラム２００との間の通信量を制御する。すなわち、サーバ・クラスタシステム１１００を構成する各サーバに対するリクエストの送出を振り分け、各サーバにリクエストを送出する。
【００２８】
このように、負荷制御プログラム４００が各サーバに対するリクエストの配分を決定することによって負荷設定部が構成される。
【００２９】
サーバ・クラスタシステム１１００は、複数のサーバがネットワークによって疎結合されており、クライアント１００からは単一のサーバとして見えるシステムである。
【００３０】
このサーバ・クラスタシステム１１００には、クラスタ再構成技術が適用され、サービスを停止させることなく、システムを構成するサーバの台数を変更することができる。このクラスタ再構成技術は、サーバ・クラスタシステム１１００に新たなサーバの追加、及びサーバ・クラスタシステム１１００で動作しているサーバの削減ができる。なお、サーバ・クラスタシステム１１００に追加された新たなサーバを「追加サーバ９００」、既にサーバ・クラスタシステム内で動作しているサーバを「既存サーバ８００」と呼ぶ。
【００３１】
また、クライアント１００及びサーバ・クラスタシステム１１００には、サーバ・クラスタシステム１１００のサーバやサービスの構成の状態を取得し管理する、管理サーバ６

００が接続されている。

【００３２】

サーバ・クラスタシステム１１００は、管理サーバ６００とエージェント・プログラム１０００とによってクラスタ再構成技術が実装されている。具体的には、エージェント・プログラム１０００は、サーバ・クラスタシステム１１００を構成する全てのサーバに備えられ動作している。管理サーバ６００とエージェント・プログラム１０００とは相互に通信しながらサーバ・クラスタシステム１１００の構成を管理する。エージェント・プログラム１０００は、サーバ・クラスタシステム１１００を構成する各サーバの負荷を計測し、定期的に管理サーバ６００に報告する。管理サーバ６００は、エージェント・プログラム１０００より報告を受け収集した負荷情報を分析し、サーバ・クラスタシステム１１００へのサーバの追加、又はサーバ・クラスタシステム１１００からのサーバ削減を決定する。

【００３３】

なお、管理サーバ６００を設けずに、管理サーバ６００の機能を、サーバ・クラスタシステム１１００に組み込んだり、クライアント１００に組み込んでもよい。

【００３４】

本発明の第１の実施の形態では、サーバ・クラスタシステム１１００に新規に追加された追加サーバ９００へのリクエスト送出量を、追加した当初（直後）は既存サーバ８００に比べて低く抑え、その後は段階的に増やしていくように制御する。追加サーバ９００がサーバ・クラスタシステム１１００に追加された直後は、追加サーバ９００に備えられているキャッシュメモリ等に充分なキャッシュが蓄えられておらず、追加サーバ９００の処理能力は既存サーバ８００と比較すると低いので、既存サーバ８００と同等のリクエストが送出されると、未処理リクエストの長い待ち行列が発生してしまう可能性がある。そのため、追加サーバ９００がサーバ・クラスタシステム１１００に追加された直後は、クライアント１００からのリクエスト送出量を低く抑えるように制御する。その後、所定の時間が経過すればキャッシュメモリ等に充分なキャッシュが蓄えられ、既存サーバ８００と同様の処理を行うことができるようになる。

【００３５】

この制御は、負荷制御関数４１１を適切に定義することで実現できる。すなわち、追加された直後のサーバの重みを、初めは小さな値に、時間経過と供に段階的に大きな値になるように算出する負荷制御関数４１１を定義する。

【００３６】

本発明のクライアント・サーバシステムの実施形態の一例として、３層Ｗｅｂシステムが挙げられる。３層Ｗｅｂシステムは、Ｗｅｂサーバ、アプリケーション（ＡＰ）サーバ、データベース（ＤＢ）サーバが、それぞれサーバ・クラスタシステムを構成し、かつ、それらのサーバ・クラスタシステムがＷｅｂサーバ、ＡＰサーバ、ＤＢサーバの順に層構造をなすシステムである。３層Ｗｅｂシステムでは、サービス利用者からのリクエストは、階層構造に従って、Ｗｅｂサーバ→ＡＰサーバ→ＤＢサーバの順に流れ作業で処理される。この３層Ｗｅｂシステムでは、サービス利用者が使用する計算機とＷｅｂサーバの間、例えば、ＷｅｂサーバとＡＰサーバの間、ＡＰサーバとＤＢサーバとの間がクライアントとサーバの関係となる。具体的には、ＷｅｂサーバとＡＰサーバとの間では、ＷｅｂサーバがクライアントでありＡＰサーバがサーバである。また、ＡＰサーバとＤＢサーバとの間では、ＡＰサーバがクライアントであり、ＤＢサーバがサーバである。

【００３７】

また、クライアント１００がＷｅｂサーバであり、Ｗｅｂサーバ・プログラムとしてＡｐａｃｈｅを用い、サーバ・クラスタシステム１１００がＡＰサーバであり、ＡＰサーバ・プログラムとしてＴｏｍｃａｔを用いた場合、負荷分散機能３００は、Ａｐａｃｈｅに組み込まれるＴｏｍｃａｔの負荷分散モジュールとなる。

【００３８】

次に、負荷制御プログラム４００の詳細について説明する。

【００３９】
図２は、負荷制御プログラム４００構造及び処理の流れを示すブロック図である。
【００４０】
負荷制御プログラム４００は、サーバ性能表４０３と負荷重み表４０５の二つのデータをテーブルとして持つ。また、サーバ性能表４０３を作成するための機能である台数検出機能４０１及び性能検出機能４０２を持つ。更に、サーバ性能表４０３から負荷重み表４０５を作成する機能である負荷重み計算機能４０４を持つ。
【００４１】
４０３は、サーバ性能表の一例を示している。サーバ性能表４０３には、サーバ・クラスタシステム１１００を構成するサーバ毎にエントリが作成される。各エントリには、サーバ性能表４０３内でのエントリ管理番号としてサーバ毎に付されたサーバ番号（サーバ＃）、サーバにアクセスを行うためのＵＲＬ（Uniform Resource Locator）情報（ＵＲＬ）、サーバの性能を示す複数のパラメータ（Ｐ１～ＰＮ）、サーバが新たに追加された時刻（ｔ０）が記録されている。
【００４２】
サーバの性能を示すパラメータの例としては、サーバのプロセッサの個数、プロセッサの動作周波数、主記憶搭載量等のシステム稼動中には変更が行われない静的パラメータ、及び、サーバ内でのキャッシュヒット率、サーバ内でのリクエスト待ち行列長等のシステム稼動中に処理の負荷状況に応じて変更される動的パラメータの２種類が記録される。
【００４３】
４０５は、負荷重み表の一例を示している。負荷重み表４０５は、サーバ性能表４０３と同じく、サーバ・クラスタシステム１１００を構成するサーバ毎にエントリが作成され記録されている。各エントリには、負荷重み表４０５内でのエントリ管理番号としてサーバ毎に付されたサーバ番号（サーバ＃）、サーバにアクセスを行うためのＵＲＬ、サーバ性能の評価値である重みが記録されている。
【００４４】
台数検出機能４０１及び性能検出機能４０２の役割は、サーバ性能表４０３のエントリを作成し記録することである。台数検出機能４０１は、サーバ性能表４０３のエントリを新規に作成し、サーバの静的パラメータを記録する。性能検出機能４０２はサーバの動的パラメータを取得し記録する。
【００４５】
負荷重み計算機能４０４は、所定のタイミングに動作し、サーバ性能表４０３を基に負荷重み表４０５の内容を更新する。具体的には、負荷重み計算機能４０４は、一定時間間隔で定期的に動作し、サーバ性能表４０３から各サーバの性能パラメータを読み込んで、読み込んだ性能パラメータを負荷制御関数４１１に入力する（メッセージ４０８）。負荷制御関数４１１は読み込んだ各サーバの性能パラメータから各サーバの重みを計算し、計算した重みを負荷重み表４０５に記録する（メッセージ４０９）。
【００４６】
なお、この負荷制御関数４１１は自由に設定し記述することが可能であり、サーバ性能の動的パラメータを入力値とした複雑な多次元関数として定義することもできる。例えば、キャッシュのヒット率が低い場合はリクエスト送出量を低くする関数、リクエストの待ち行列長が高い場合にはリクエストの送出量を低くする関数等を負荷制御関数４１１として設定する。また、複数の負荷制御関数４１１を予め設定しておき、条件によって複数の負荷制御関数４１１を切り替えて使い分けることもできる。
【００４７】
この負荷制御関数４１１の例を図３及び図４に示す。図３の例では、負荷制御関数４１１を、サーバが新たに追加されてからの時間（サーバ性能表４０３のパラメータ“ｔ０”）を入力値として関数を定義している。この負荷制御関数には、サーバが追加されてから所定の時間が経過するまではリクエスト送出量を時間に比例して増加させ、所定の時間が経過した後は入力値（経過時間）に関係なくリクエスト送出量を一定とする関数が定義さ

れている。

【００４８】

図４の例では、負荷制御関数４１１を、待ち行列長を入力値として関数を定義している。図４によると、待ち行列長とリクエスト送出量とが反比例の関係となるような関数が定義されている。すなわち、待ち行列長が長い間はリクエスト送出量を減少させ、待ち行列長が短くなるとリクエスト送出量を増加させる。

【００４９】

図５は、負荷制御プログラム４００の台数検出機能４０１の処理を示すフローチャートである。なお、図５では、サーバ・クラスタシステム１１００に新たにサーバ（追加サーバ９００）が追加された場合を例として示している。

【００５０】

まず、台数検出機能４０１は通常は待機状態にある。ここで、管理サーバ６００からサーバ・クラスタシステム１１００の構成（サーバの数等）が変更されたことの通知を受信すると（処理１２０１）、該通知から追加サーバ９００に関する情報を取得する（処理１２０２）。

【００５１】

次に、管理サーバ６００から受信した追加サーバ９００に関する情報から、追加サーバ９００のサーバ名を取得する（処理１２０３）。次に、取得したサーバ名を用いて、サーバ性能表４０３内に該追加サーバ９００のエントリが存在するかを判定する（処理１２０４）。エントリが無いと判定した場合は、追加サーバ９００に対する一意なサーバ番号を新規に生成する。そして、生成したサーバ番号のエントリをサーバ性能表４０３に新たに作成し（処理１２０５）、処理１２０６に進む。既にエントリがあると判定した場合は、エントリの新規作成は行わず処理１２０６に進む。

【００５２】

処理１２０６では、受信した追加サーバ９００に関する情報から、追加サーバ９００のＵＲＬ、追加サーバ９００の静的なパラメータを抽出して取得し、サーバ性能表４０３の該当するサーバ番号のエントリに書き込んで記録する。その後、再び待機状態に戻り、構成が変更されたことの通知を待機する。

【００５３】

このように、台数検出機能４０１が追加サーバの数が変更されたことを検出することによって台数検出部が構成される。

【００５４】

図６は、負荷制御プログラム４００の、性能検出機能４０２の処理を示すフローチャートである。

【００５５】

性能検出機能４０２は所定のタイミングに動作し、サーバ性能表４０３の各エントリの動的パラメータの値を更新する。まず、サーバ性能表４０３内のエントリを調査し、現在サーバ性能表４０３に記録されているサーバ名を取得する（処理１３０１）。次いで、サーバ性能表４０３に記録されている各サーバと通信し、動的パラメータに関する性能情報を取得する（処理１３０２）。次に、取得した性能情報をサーバ性能表４０３に記録する（処理１３０３）。そして、この処理１３０１から処理１３０３の処置をエントリの数だけ繰り返す。

【００５６】

このように、性能検出機能４０１が、各サーバの状態である動的パラメータを取得することによって状態取得部が構成される。

【００５７】

図７は、負荷重み計算機能４０４の処理を示すフローチャートである。

【００５８】

負荷重み計算機能４０４は、所定のタイミング（例えば、タイマのカウント等によって定期的）にサーバ性能表４０３を参照し、サーバ性能表４０３から所定のエントリのサー

バの性能パラメータを取得する（処理１４０１）。次に、取得した性能パラメータを入力値として負荷制御関数４１１に入力し、該エントリのサーバの重みを計算する（処理１４０２）。次に、計算した重みを、負荷重み表４０５の当該サーバのエントリに書き込み記録する（処理１４０３）。次に、サーバ番号を加算することで次のエントリを設定する（処理１４０４）。このようにすることで、サーバ性能表４０３のサーバ番号の順に、全てのエントリに対する重みが算出される。

【００５９】

なお、この図７の負荷重み計算機能４０４の処理の実行タイミングは、定期的に行う以外に、台数検出機能４０１又は性能検出機能４０２がサーバ性能表４０３のエントリを更新したことを契機に行うことができる。例えば、台数検出機能４０１が、サーバ性能表４０３のエントリを更新した後、その旨を負荷重み計算機能４０４に通知する。負荷重み計算機能４０４は、この通知を契機として図７のフローチャートに従って重みを算出し、負荷重み表４０５を更新する。性能検出機能４０２がサーバ性能表４０３を更新した場合も、負荷重み計算機能４０４に対して同様の通知を行うことで重みを算出し、負荷重み表４０５を更新する。

【００６０】

次に、負荷分散機能３００によって、クライアントプログラム２００とサーバ・クラスタシステム１１００との間で行われる負荷分散の処理について説明する。

【００６１】

まず、コネクション・プールについて説明する。

【００６２】

コネクション・プールとは、計算機間通信の高速化技術である。一般に、計算機間で通信を行う場合、「コネクション」と呼ばれる通信路を確立する必要がある。なお、確立されたコネクションは通信が終了すると破棄される。このコネクションを確立する処理には時間を要するため、通信の度にコネクションを確立する処理をしていては通信効率が低下する。コネクション・プールは、一度確立したコネクションを、使用後（通信終了後）にも廃棄せずにコネクションの状態を記憶（プール）する。プールされたコネクションは、再度、同一経路での通信が行われる際に再利用することで、再度コネクションを確立する処理を省略することができ、通信効率を向上することができる。

【００６３】

図８は、三台のサーバ（８００ａ、８００ｂ、８００ｃ）で構成されたサーバ・クラスタシステム１１００にクライアントプログラム２００がリクエストを送出する場合のコネクション・プールの実装例を示す。

【００６４】

コネクション配分機能３０１は、各々のサーバにコネクション・プール３０４を作成し、その管理を行う。

【００６５】

図８は、クライアント１００からサーバ８００ａへは、五つのコネクション・プール３０４ａのうち三つのコネクション（図８中の網掛け部分）を使用し、クライアント１００からサーバ８００ｂへは五つのコネクション・プール３０４ｂのうち一つのコネクションを使用し、クライアント１００からサーバ８００ｃへは五つのコネクション・プール３０４ｃのうち二つのコネクションを使用している状況を表している。

【００６６】

なお、図８では全てのサーバに対して同じ数のコネクション・プールを設けているが、プール可能なコネクションの数はサーバ毎に異なっていても構わない。

【００６７】

コネクション配分機能３０１は、コネクション管理表３０２に、プールしているコネクションの数と現在使用中のコネクションの数とを、コネクション・プール３０４毎に記録する（メッセージ３０３）。

【００６８】

クライアントプログラム２００は、サーバ・クラスタシステム１１００にリクエストを送出する際、通信に必要なコネクションをコネクション・プールから取得して使用するために、まずコネクション配分機能３０１に対して、コネクションの割当てを要求する（メッセージ２０１）。要求を受けたコネクション配分機能３０１は、まずコネクション管理表３０２を参照し（メッセージ３０３）、どのコネクション・プール３０４からコネクションを取得するかを決定する（図８では、コネクション・プール３０４ｂから取得）。次に、取得したコネクションをクライアントプログラム２００に割当て、その旨をクライアントプログラム２００に送信する（メッセージ２０２）。クライアントプログラム２００は、割当てられたコネクションを使用してサーバ８００ｂと通信を行う。クライアントプログラム２００は、通信が終了すると、コネクションの使用が終了した旨のメッセージ２０３をコネクション配分機能３０１に送信して、コネクションをコネクション配分機能３０１に返却する。コネクション配分機能３０１は、このメッセージを受け取って、使用中のコネクションを未使用のコネクションに変更して、コネクション管理表３０２を更新する。

【００６９】

図９は、負荷分散機能３００に前述したコネクション・プールを用いた場合の、負荷分散機能のデータ構造、及びデータの処理を示すブロック図である。

【００７０】

負荷分散機能３００は、クライアントプログラム２００とサーバ・クラスタシステム１１００との間の負荷分散を行う。負荷分散機能３００は、コネクション配分機能３０１とコネクション管理表３０２を保持している。

【００７１】

クライアントプログラム２００は、サーバ・クラスタシステム１１００にリクエストを送出する際に、通信に必要なコネクションの割当てを要求する（メッセージ２０１）。要求を受けたコネクション配分機能３０１は、コネクション管理表３０２を参照して（メッセージ３０３）、どのコネクション・プールからコネクションを取得するかを決定する。

【００７２】

コネクション管理表３０２は、図９に示すように、サーバ番号（サーバ#）、サーバにアクセスするためのＵＲＬ、プール可能なコネクション数の最大の値（最大コネクション）、現在使用されているコネクションの数（使用中コネクション）を一覧として保持している。

【００７３】

コネクション配分機能３０１は、このコネクション管理表３０２を参照して、どのサーバにいくつのコネクションを割当てるかを決定し、結果をコネクション管理表３０２に記録して反映する。クライアントプログラム２００は、割当てられたコネクションを使用してサーバにリクエストを送出する。

【００７４】

図１０は、負荷分散機能３００の処理を示すフローチャートである。

【００７５】

負荷分散機能３００は、クライアントプログラム２００からコネクション割当ての要求を受信すると（処理１５０１）、コネクション管理表３０２を参照し、現在のコネクション割当て状況に関する情報を取得する（処理１５０２）。次に、負荷重み表４０５を参照し、各サーバの重みを取得する（処理１５０３）。なお、処理１５０２と処理１５０３の順序は逆でもよい。

【００７６】

次に、コネクション配分機能３０１が、取得した現在のコネクション割当て状況とサーバ毎の重みとを基に、重み付きラウンドロビン等の負荷分散アルゴリズムに従って、どのサーバにいくつのコネクションを割当てるかを選定する（処理１５０４）。次に、選定したコネクションを、クライアントプログラム２００に通知し、各サーバに対するコネクションを割当てる（処理１５０５）。

【００７７】
このように、コネクション・プールを使用することで、サーバ・クラスタシステム１１００を構成する各サーバの負荷に応じて、各サーバのコネクションの割当てを決定する。各サーバに割当てたコネクションの数に従って、リクエスト送出の割合が決まる。
【００７８】
以上説明したように、本発明の第１の実施の形態のクライアント・サーバシステムでは、クライアント１００がリクエストを送出するサーバ・クラスタシステム１１００の構成が変更され、新たに追加サーバ９００が追加されたときは、該追加サーバ９００に対するリクエストの送出量を、既存サーバ８００と比較して少なくする。このようにすることで、追加サーバ９００の、未処理リクエストの長い待ち行列の発生を抑えることができ、サーバ・クラスタシステム１１００全体としての処理の効率を高めることができる。
【００７９】
なお、第１の実施の形態では、性能検出機能４０２が取得する動的パラメータの例として、キャッシュヒット率、待ち行列長を挙げたが、その他に、メモリ使用量、スワップ回数等のメモリに関連する情報、ＣＰＵ使用率等のＣＰＵに関連する情報、物理ディスクへの入出力量等の入出力に関連する情報、ネットワーク通信量等のネットワークに関連する情報等を動的パラメータに使用して、リクエスト送出量を制御してもよい。
【００８０】
また、負荷制御関数４１１への入力値として、動的パラメータの値をそのまま入力したが、値の変化量を負荷制御関数４１１に入力してもよい。例えば、取得した待ち行列長の値が１０から３０に変化した場合、入力値を３０とするのではく、変化量の２０を入力値とする。変化量を入力することで、待ち行列長が短期間に急激に上昇した場合に、重みを小さくする制御が可能となる。なお、変化量を算出するためには、以前に取得した値を、負荷重み表４０５又は別の表に記録しておく必要がある。
【００８１】
また、性能検出機能４０２は、サーバ・クラスタシステム１１００に属するサーバから直接性能情報５００を取得したが、その他の取得方法として、管理サーバ６００がサーバ・クラスタシステム１１００に属するサーバから性能情報５００を取得し、その後、性能検出機能４０２へ性能情報５００をまとめて送信してもよい。
【００８２】
また、負荷分散機能３００及び負荷制御プログラム４００は、各クライアント１００内に実装されているが、負荷分散装置を別に設け、負荷分散装置によって、複数のクライアント１００からのリクエストを集約し、サーバ・クラスタシステム１１００の各サーバにリクエストを振り分けるように実装してもよい。
【００８３】
次に、本発明の第２の実施の形態について説明する。
【００８４】
第２の実施の形態は、本発明をストレージ・システム（ストレージ装置）に適用した場合の実施形態であり、クライアント、ディレクトリサーバ・クラスタシステム及びストレージによって、サーバ・クライアントシステムが構成されている。
【００８５】
図１１は、本発明の第２の実施の形態のストレージ・システムの構成を示したブロック図である。
【００８６】
本実施の形態のストレージ・システムは、ディレクトリサーバ・クラスタ２６００が、クライアント２０００にファイルサービスを提供する。
【００８７】
クライアント２０００は、ディレクトリサーバ・クラスタ２６００に対してファイルサービスを要求する。ディレクトリサーバ・クラスタ２６００は、複数のディレクトリサーバ（既存ディレクトリサーバ２１００及び追加ディレクトリサーバ２２００）がネットワ

ークによって疎結合されており、クライアント２０００からは単一のディレクトリサーバとして見えるクラスタシステムである。

【００８８】

ファイルの実体（データ）はストレージ２３００に格納されている。ディレクトリサーバ・クラスタ２６００とストレージ２３００とは、ＳＡＮ（Storage Area Network）２５００によって接続されている。ディレクトリサーバ・クラスタ２６００とＳＡＮ２５００とはネットワーク２５０１を介して接続され、ストレージ２３００とＳＡＮ２５００とはネットワーク２５０２を介して接続されている。また、クライアント２０００とディレクトリサーバ・クラスタ２６００とは、サーバ間ネットワーク２４００によって接続されている。

【００８９】

一般的なファイルシステムは、ファイルの格納場所を指示する「ディレクトリ情報」と、ディレクトリ情報によって指示される「ファイルの実体」とからなり、ディレクトリ情報とファイルの実体（データ）は同一の記憶装置に格納される。

【００９０】

本実施の形態のストレージ・システムでは、ディレクトリ情報はディレクトリサーバ・クラスタ２６００に格納され、ファイルの実体はストレージ２３００に格納される。そのため、ディレクトリ情報には、ファイルが格納されているストレージを示す情報と、該ストレージ内での格納位置を示す情報が記録される。

【００９１】

これら複数のディレクトリサーバ（既存ディレクトリサーバ２１００及び追加ディレクトリサーバ２２００）はクラスタ構成をとっており、ディレクトリ情報は複数のディレクトリサーバ２２００に分散して格納される。なお、あるファイルのディレクトリ情報を格納しているディレクトリサーバ２１００を、そのファイルの「担当ディレクトリサーバ」と呼ぶ。あるファイルとそのファイルの担当ディレクトリサーバとの対応は、クライアント２０００が保持するファイル割当て管理表２００１に記録されている。なお、全てのクライアント２０００は、同一内容のファイル管理表２００１を持っている。

【００９２】

このように、クライアント２０００が、あるファイルとそのファイルの担当ディレクトリサーバとの対応を割当管理表２００１が保持することによって、割当保持部が構成される。

【００９３】

次に、クライアント２０００が出すファイル取得要求を、このストレージ・システムが処理する手順について説明する。

【００９４】

クライアント２０００は、ストレージ・システムのディレクトリサーバ・クラスタ２６００にファイル取得要求（メッセージ２４０１）を送信する。クライアント２０００がファイルを取得するためには、まず、取得を要求するファイルの担当ディレクトリサーバを特定する必要がある。クライアント２０００は、ファイルとそのファイルの担当ディレクトリサーバとの対応が記録されているファイル割当て管理表２００１を参照し、取得を要求するファイルの担当ディレクトリサーバを特定する。次に、特定した担当ディレクトリサーバにファイル取得要求を送信する。

【００９５】

ディレクトリサーバ２１００は、クライアント２０００からのファイル取得要求を受け取ると、ディレクトリ情報を参照して、該ファイルが格納されているストレージ２３００を特定する。次に、特定したストレージ２３００にファイルを要求する。ファイルの要求を受け取ったストレージ２３００はファイルを送信し、ディレクトリサーバ２１００はファイルを取得する。取得したファイルは、要求を送信したクライアント２０００に対して送信される。

【００９６】

ここで、ディレクトリサーバ２１００のキャッシュについて説明する。

【００９７】

一般に、ディレクトリサーバはキャッシュメモリ等のキャッシュを持っている。ディレクトリサーバ２１００は、一度ストレージ２３００から取得したファイルをメモリ上のキャッシュ領域に格納する。以降、ディレクトリサーバ２１００は、同一のファイルの取得要求に対しては、キャッシングしたファイルを送信し、ストレージ２３００からファイルを取得する処理を省略する。このようにすることで、ディレクトリサーバ２１００からファイルを取得する処理の効率が向上する。

【００９８】

ここで、第１の実施の形態のようにディレクトリサーバ・クラスタ２６００を構成するディレクトリサーバの数を動的に変更する場合を説明する。

【００９９】

追加ディレクトリサーバ２２００がディレクトリサーバ・クラスタ２６００に追加された場合、既存ディレクトリサーバ２１００が担当しているストレージ２３００のファイルのうち、いくつかのファイルの担当が追加ディレクトリサーバ２２００に移管される。すなわち、いくつかのファイルの担当ディレクトリサーバが既存ディレクトリサーバ２１００から追加ディレクトリサーバ２２００に変更される。

【０１００】

ストレージ２３００のファイルの担当ディレクトリサーバが変更されると、これに伴って、クライアント２０００が持つファイル割当て管理表２００１の内容が変更される。具体的には、例えば、ファイル“Ｆｉｌｅ１”の担当ディレクトリサーバがサーバ“ＳＲＶ１”からサーバ“ＳＲＶ２”に変更される。この場合、全てのクライアント２０００が持つファイル割当て管理表２００１の内容が更新される。このファイル割当て管理表２００１の更新は、第１の実施の形態と同様に管理サーバを設け、管理サーバから各クライアントに通知してもよいし、ディレクトリサーバ２１００からクライアント２０００に対して直接通知してもよい。

【０１０１】

次に、この新たに追加された追加ディレクトリサーバ２２００のキャッシュの状態について説明する。追加ディレクトリサーバ２２００をディレクトリサーバ・クラスタ２６００に追加し、いくつかのファイルの担当ディレクトリサーバ２１００が追加ディレクトリサーバ２２００に変更された時点では、追加ディレクトリサーバ２２００のキャッシュにはファイルが一切キャッシングされていない。そのため、担当ディレクトリサーバが変更されたファイルにファイルの取得要求があった場合には、要求されたファイルを一度ストレージ２３００に要求する必要があり、キャッシュが効果的に働かない。そのため、該ファイル取得要求に対する返答には多大な時間がかかる。

【０１０２】

したがって、多数のファイルの担当ディレクトリサーバを一度に追加ディレクトリサーバ２２００に変更した場合は、追加ディレクトリサーバ２２００での処理が滞り、結果として、ディレクトリサーバ・クラスタ２６００からクライアント２０００への返答が全体的に悪化する。このようなシステムの性能悪化を防止するために、追加ディレクトリサーバ２２００が担当するファイルを一時に一度に増やすのではなく、徐々に増やす必要がある。

【０１０３】

既存ディレクトリサーバ２１００から追加ディレクトリサーバ２２００に、ファイルの担当を徐々に移管する方法について説明する。この移管は、クライアント２０００が持つファイル割当て管理表２００１の変更に、非特許文献１のようにハッシュ関数を用いることで実現できる。

【０１０４】

図１２はファイル割当て管理表２００１を変更する方法を模式的に示した説明図であり、ファイルの担当を徐々に移管する方法の一例を示す。このファイル割当て管理表２００

１は、ハッシュ関数２００２と、サーバ名変換表２００３から構成される。

【０１０５】

クライアント２０００は、ファイル取得要求を送信する際に、まず、ファイル名をハッシュ関数２００２に入力し、ファイル名をハッシュ値に変換する。なお、ここで変換されたハッシュ値の最大値は、ディレクトリサーバの総数よりも充分大きな値になるようにハッシュ関数を設定する。次に、変換したハッシュ値からサーバ名変換表２００３を引くことで参照し、ハッシュ値とディレクトリサーバ名との対応を取得し、そのファイルの担当ディレクトリサーバ名を特定する。

【０１０６】

ここで、追加ディレクトリサーバ２２００が担当するファイルを徐々に増やすには、サーバ名変換表２００３を徐々に変更する操作で実現できる。具体的には、例えば、既存ディレクトリサーバ２１００が担当していたファイル“Ｆｉｌｅ１”の担当ディレクトリサーバを追加ディレクトリサーバ２２００へ変更する場合、ファイル“Ｆｉｌｅ１”のハッシュ値に対応するサーバ名変換表２００３のエントリに記録されている既存ディレクトリサーバ２１００を追加ディレクトリサーバ２２００に変更する。この変更操作を、一度に複数のハッシュ値に対して行うのではなく、段階的に徐々に行う。

【０１０７】

この、既存ディレクトリサーバ２１００から追加ディレクトリサーバ２２００に、担当ディレクトリサーバを変更するファイルを増やす方法については、前述の第１の実施の形態と同様である。すなわち、図１の負荷制御プログラム４００と同等の働きをするプログラムが各クライアント２０００上で動作し、担当ディレクトリサーバが変更されるファイルの数を、負荷制御関数４１１によって算出することで徐々に増加させる。

【０１０８】

本発明の第２の実施の形態のストレージ・システムでは、前述したように、クライアント２０００がファイルの取得を要求するディレクトリサーバ・クラスタ２６００の構成が変更され、新たに追加ディレクトリサーバ２２００が追加されたときは、該追加ディレクトリサーバ２２００が担当するストレージ２３００のファイルの数を、既存ディレクトリサーバ２１００と比較して小さくする。このようにすることで、追加ディレクトリサーバ２２００での処理が滞ることを防ぐことができ、ディレクトリサーバ・クラスタ２６００全体としての処理の効率を高めることができる。

【産業上の利用可能性】

【０１０９】

本発明は、Ｗｅｂサーバ→アプリケーションサーバ→データベースサーバのように階層構造に従って処理される階層Ｗｅｂシステムにおける、前段サーバと後段サーバとの負荷分散に適用すると有用である。また、複数のディレクトリサーバが複数のストレージ２３００のディレクトリ情報を分担して保持するストレージ・システムのディレクトリサーバにおける負荷分散に適用すると有用である。

【図面の簡単な説明】

【０１１０】

【図１】本発明の第１の実施の形態のクライアント・サーバシステムの概要を表すブロック図である。

【図２】本発明の第１の実施の形態の負荷制御プログラム４００構造及び処理の流れを示すブロック図である。

【図３】本発明の第１の実施の形態の負荷制御関数４１１の例を示すグラフである。

【図４】本発明の第１の実施の形態の負荷制御関数４１１の他の例を示すグラフである。

【図５】本発明の第１の実施の形態の台数検出機能４０１の処理を示すフローチャートである。

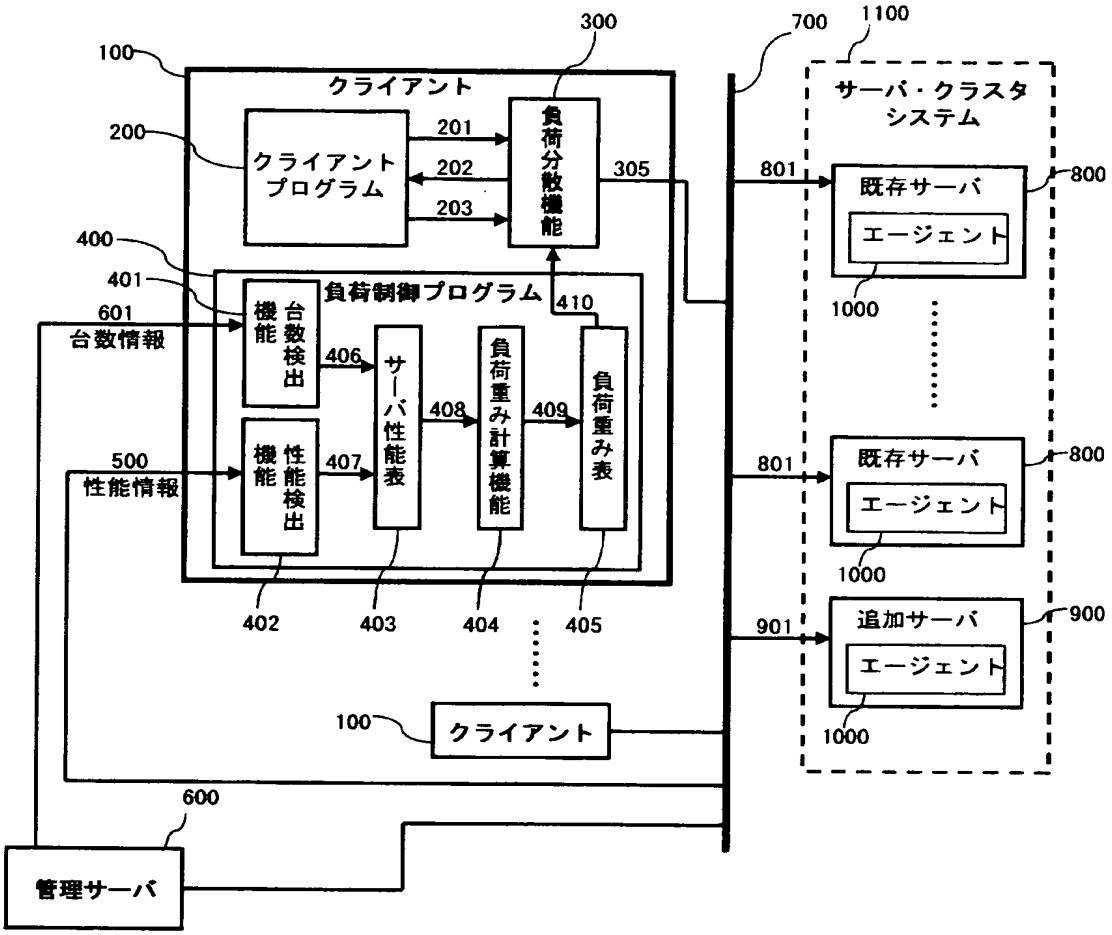
【図６】本発明の第１の実施の形態の性能検出機能４０２の処理を示すフローチャートである。

【図７】本発明の第１の実施の形態の負荷重み計算機能４０４の処理を示すフローチャートである。
【図８】コネクション・プールの実装例を示すブロック図である。
【図９】本発明の第１の実施の形態の負荷分散機能３００データ構造、及びデータの処理を示すブロック図である。
【図１０】本発明の第１の実施の形態の負荷分散機能３００の処理を示すフローチャートである。
【図１１】本発明の第２の実施の形態のストレージ・システムの構成を示したブロック図である。
【図１２】ファイル割当て管理表２００１を変更する方法を模式的に示した説明図である。

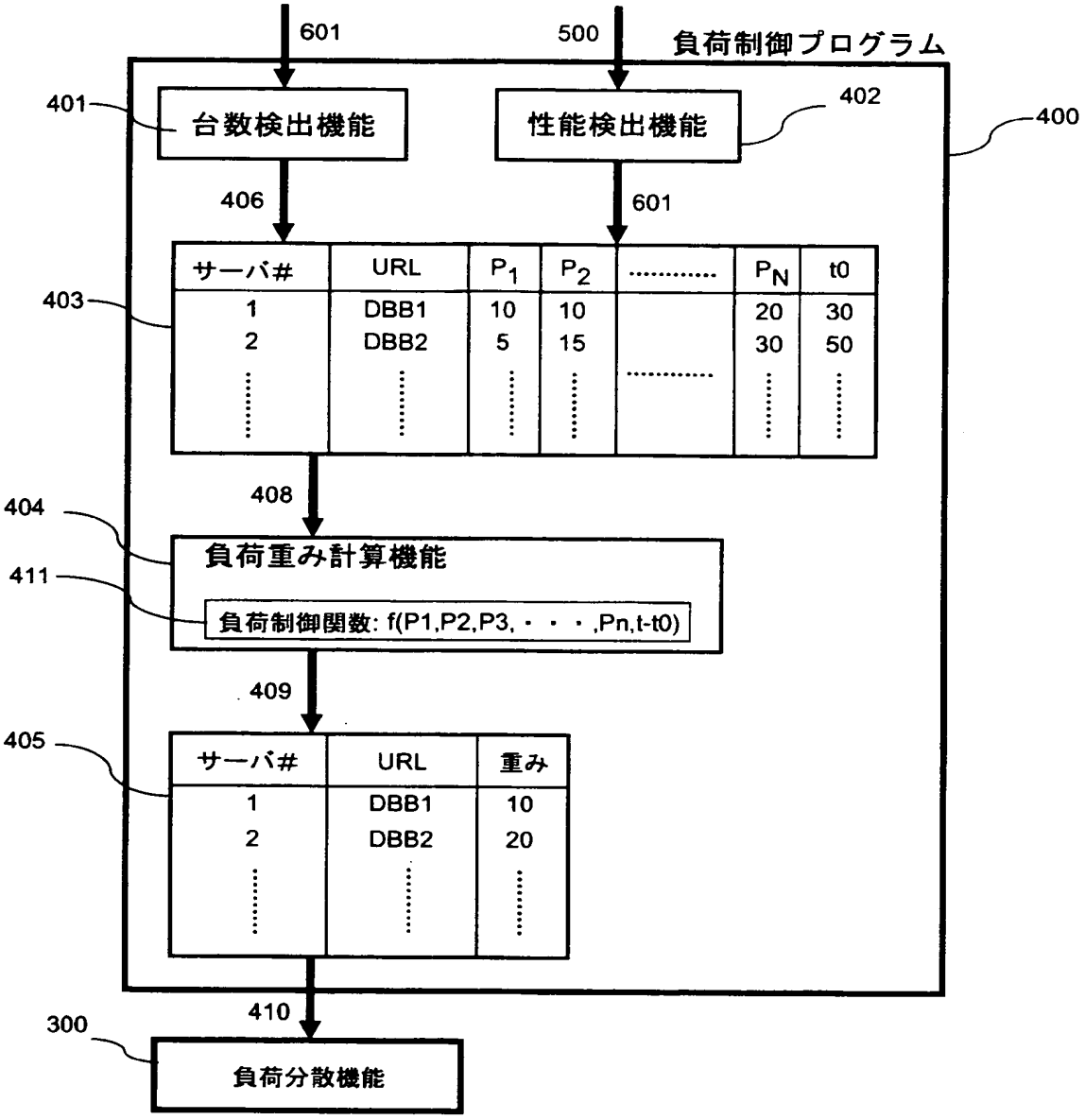
【符号の説明】

【０１１１】
１００　クライアント
２００　クライアントプログラム
３００　負荷分散機能
３０１　コネクション配分機能
３０２　コネクション管理表
４００　負荷制御プログラム
４０１　台数検出機能
４０２　性能検出機能
４０３　サーバ性能表
４０４　負荷重み計算機能
４０５　負荷重み表
４１１　負荷制御関数
５００　性能情報
６００　管理サーバ
７００　サーバ間ネットワーク
８００　既存サーバ
９００　追加サーバ
１０００　エージェント・プログラム
１１００　サーバ・クラスタシステム
２０００　クライアント
２００１　ファイル割当て管理表
２００３　サーバ名変換表
２１００　既存ディレクトリサーバ
２１０１　エージェント
２２００　追加ディレクトリサーバ
２３００　ストレージ
２４００　ＬＡＮ
２５００　ＳＡＮ
２６００　ディレクトリサーバ・クラスタ

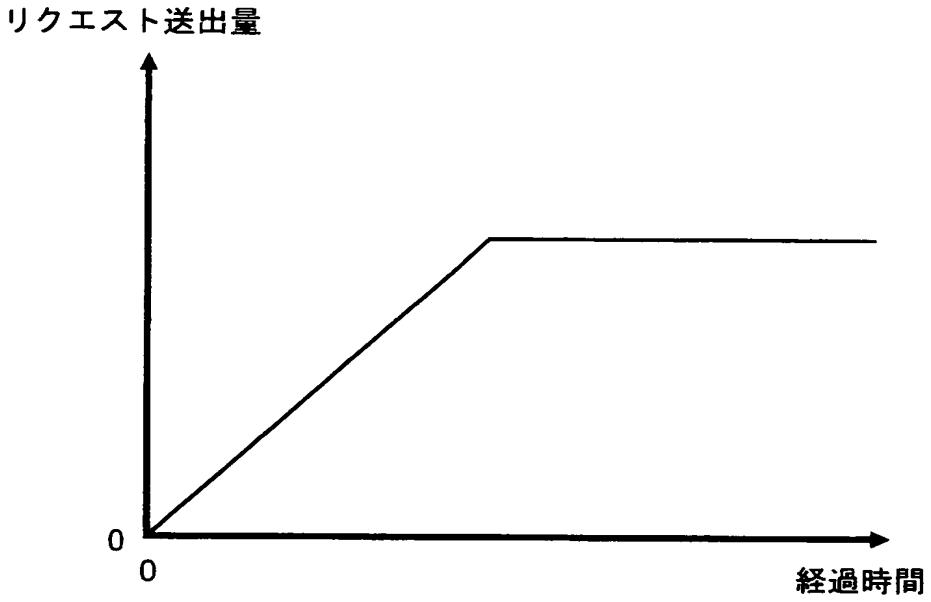
【書類名】図面
【図１】


100
クライアント
200
クライアントプログラム
201
202
203
300
負荷分散機能
305
700
1100
サーバ・クラスタシステム
801
既存サーバ
エージェント
800
1000
400
401
負荷制御プログラム
601
台数情報
台数検出機能
406
サーバ性能表
408
負荷重み計算機能
409
負荷重み表
410
500
性能情報
性能検出機能
407
402
403
404
405
801
既存サーバ
エージェント
800
1000
901
追加サーバ
エージェント
900
1000
100
クライアント
600
管理サーバ

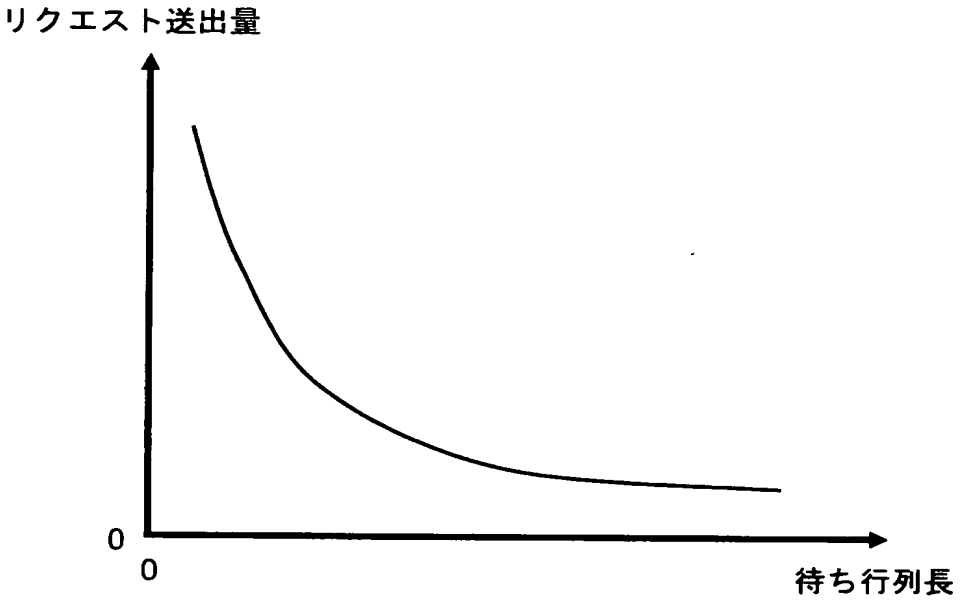
【図２】


601
500
負荷制御プログラム
401
台数検出機能
402
性能検出機能
400
406
601
403
サーバ#
URL
P1
P2
.....
PN
t0
1
DBB1
10
10
20
30
2
DBB2
5
15
30
50
408
404
負荷重み計算機能
411
負荷制御関数: f(P1,P2,P3,・・・,Pn,t-t0)
409
405
サーバ#
URL
重み
1
DBB1
10
2
DBB2
20
410
300
負荷分散機能

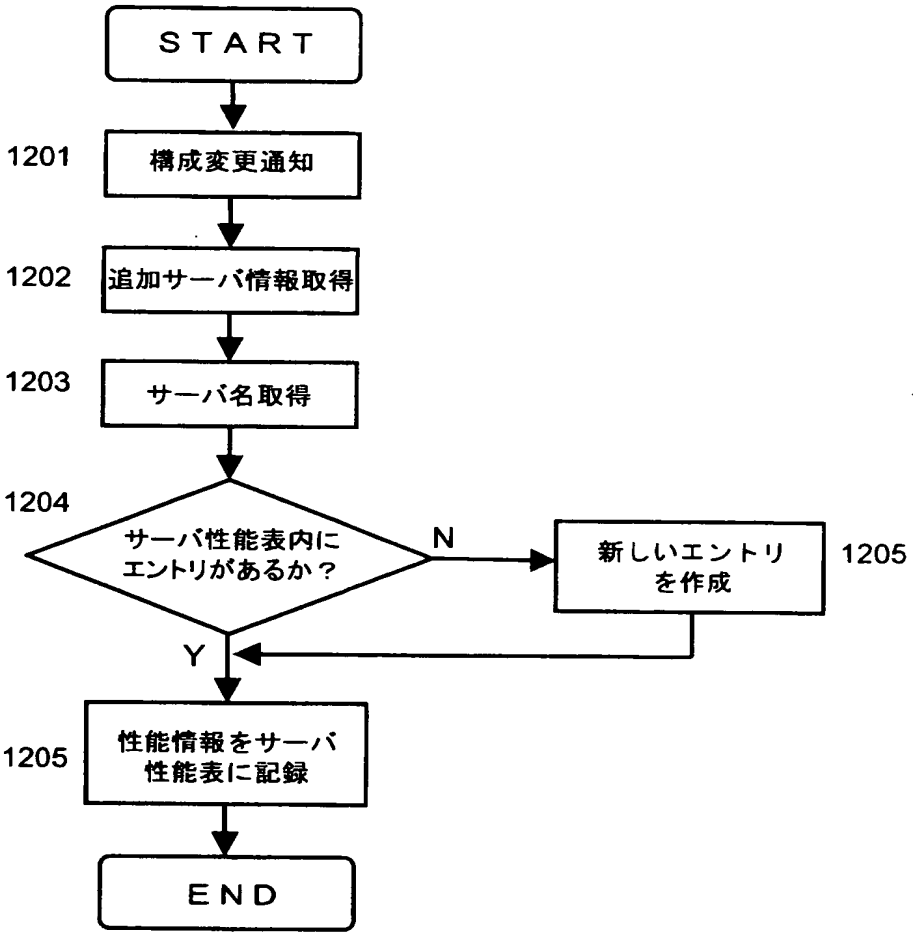
【図３】



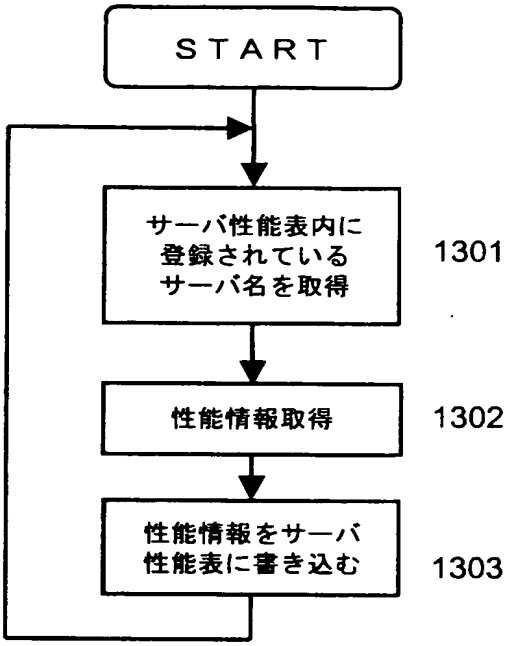
【図４】



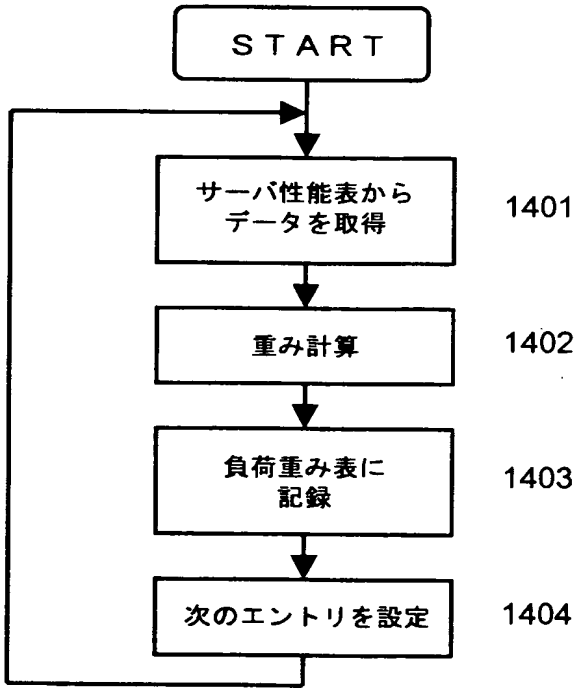
【図５】


START
1201
構成変更通知
1202
追加サーバ情報取得
1203
サーバ名取得
1204
サーバ性能表内にエントリがあるか？
N
新しいエントリを作成
1205
Y
1205
性能情報をサーバ性能表に記録
END

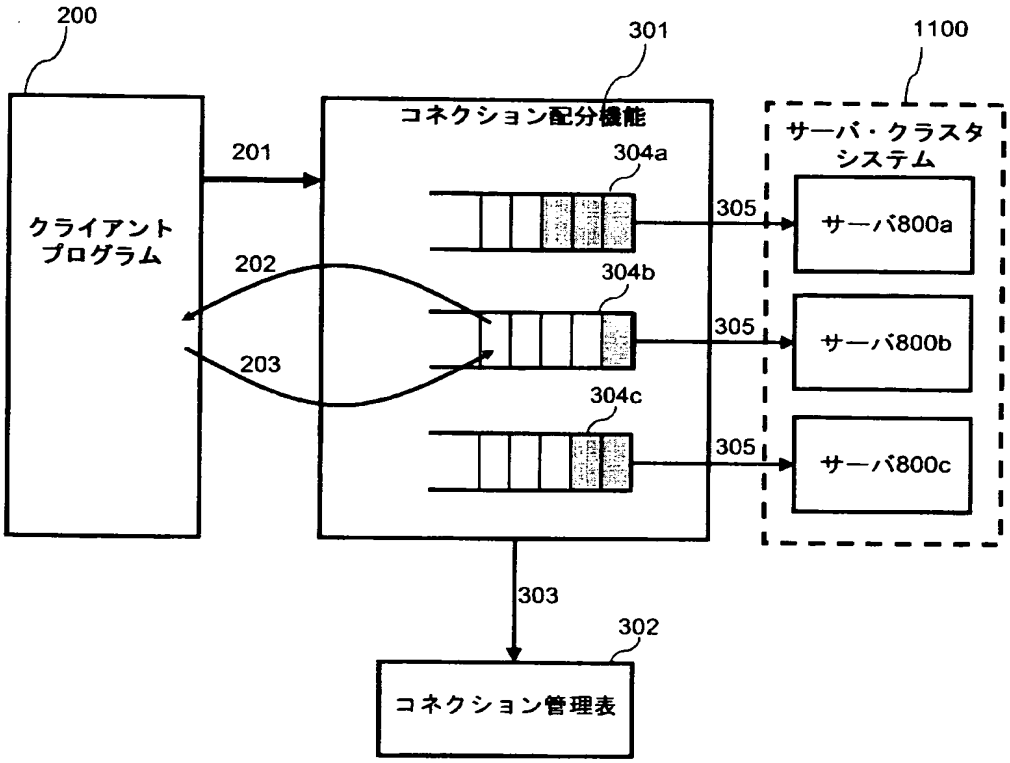
【図６】


START
サーバ性能表内に
登録されている
サーバ名を取得
1301
性能情報取得
1302
性能情報をサーバ
性能表に書き込む
1303

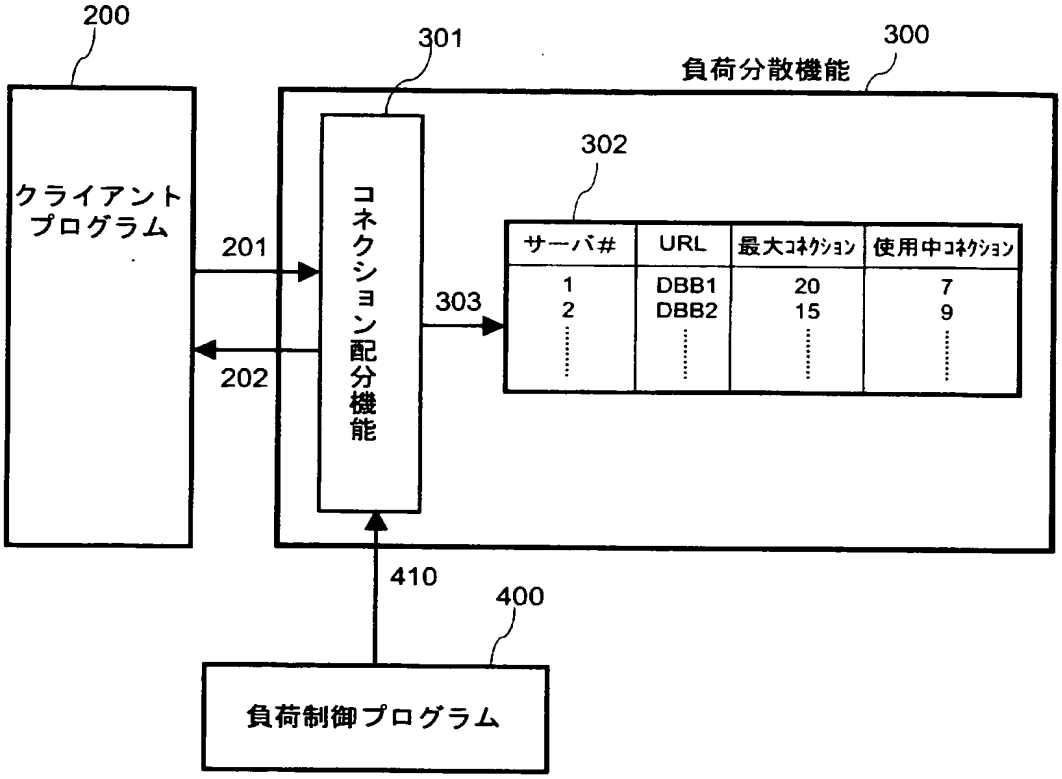
【図７】


START
サーバ性能表から
データを取得
1401
重み計算
1402
負荷重み表に
記録
1403
次のエントリを設定
1404

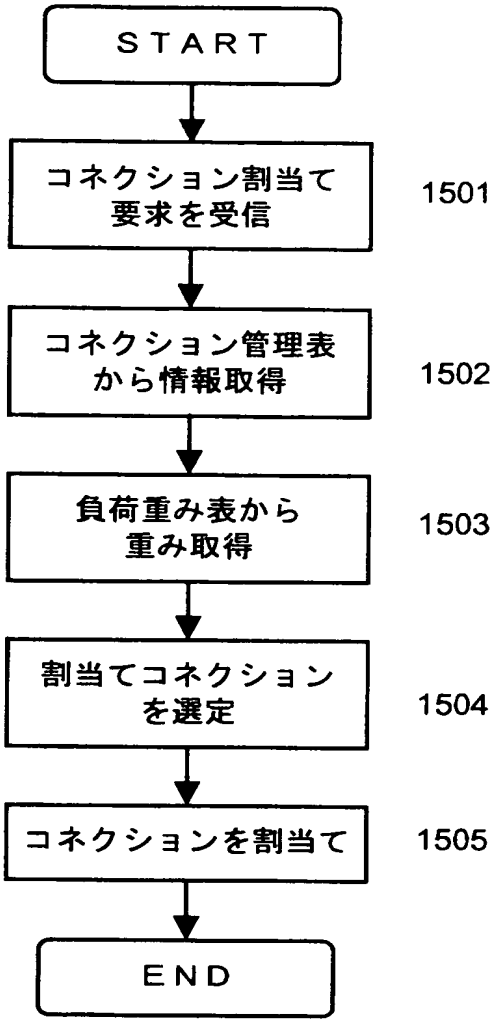
【図８】


200
301
1100
コネクション配分機能
サーバ・クラスタ
システム
クライアント
プログラム
201
304a
305
サーバ800a
202
304b
305
サーバ800b
203
304c
305
サーバ800c
303
302
コネクション管理表

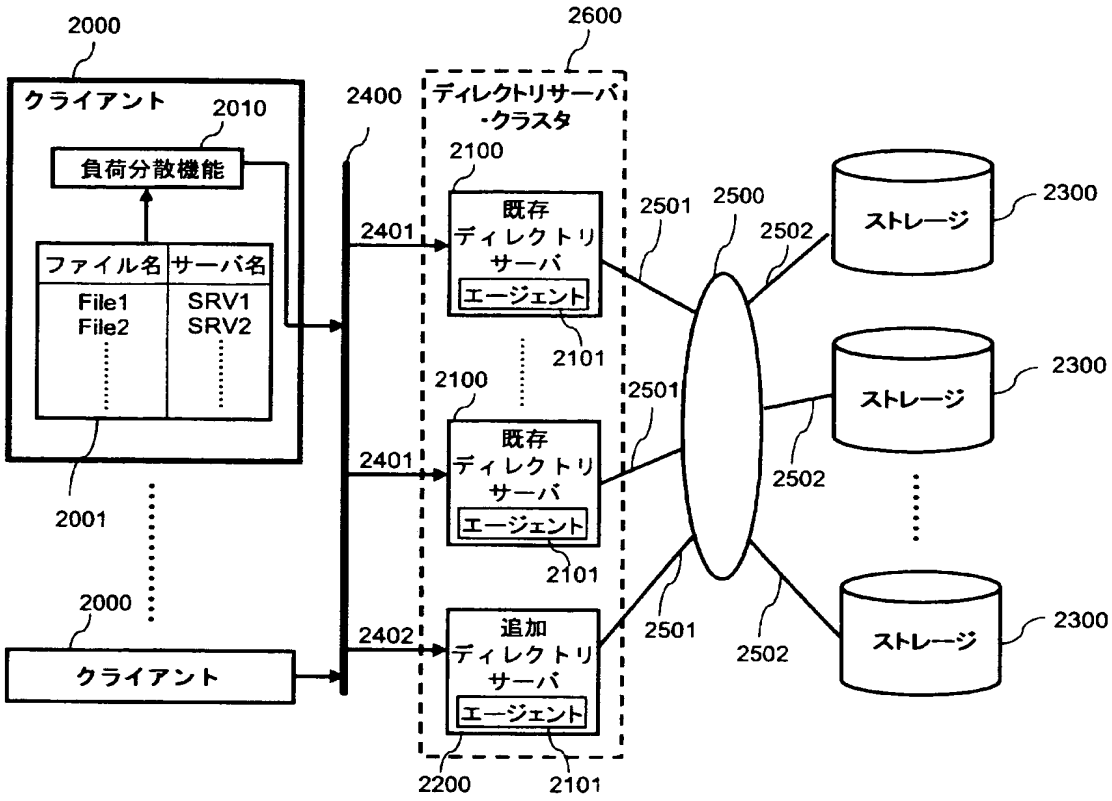
【図９】


200
301
300
負荷分散機能
クライアント
プログラム
コネクション配分機能
302
サーバ#
URL
最大ｺﾈｸｼｮﾝ
使用中ｺﾈｸｼｮﾝ
1
2
DBB1
DBB2
20
15
7
9
201
202
303
410
400
負荷制御プログラム

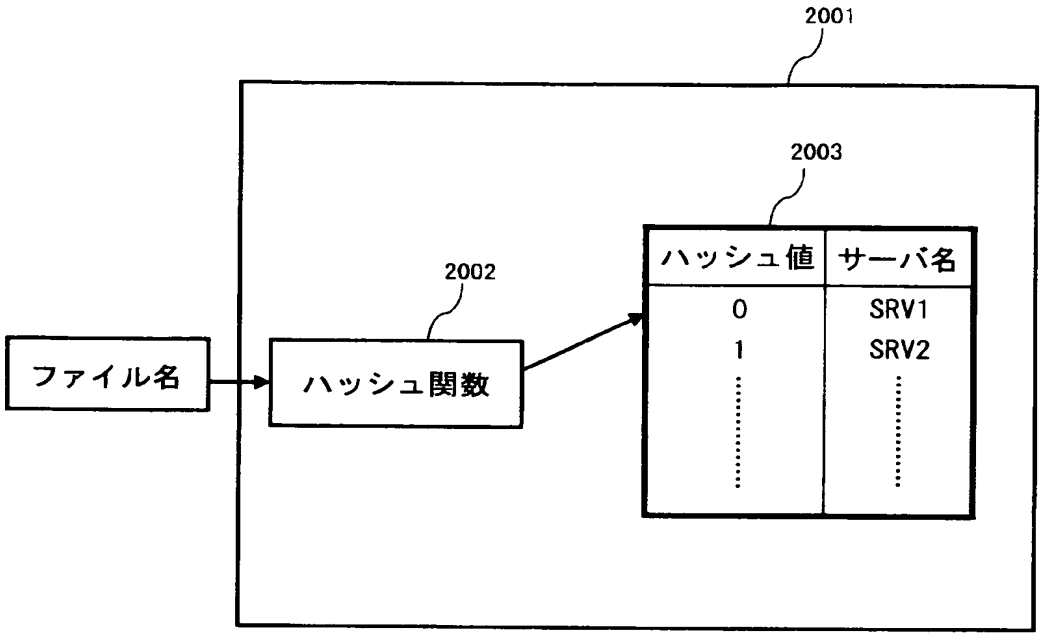
【図１０】


START
コネクション割当て要求を受信
1501
コネクション管理表から情報取得
1502
負荷重み表から重み取得
1503
割当てコネクションを選定
1504
コネクションを割当て
1505
END

【図１１】


2000
クライアント
2010
負荷分散機能
ファイル名
サーバ名
File1
File2
SRV1
SRV2
2001
2000
クライアント
2400
2600
ディレクトリサーバ
-クラスタ
2100
既存
ディレクトリ
サーバ
エージェント
2101
2401
2100
既存
ディレクトリ
サーバ
エージェント
2101
2401
2402
追加
ディレクトリ
サーバ
エージェント
2200
2101
2501
2500
2502
2501
2502
2501
2502
ストレージ
2300
ストレージ
2300
ストレージ
2300

【図１２】



【書類名】要約書
【要約】
【課題】サービス需要の増減に応じてサーバ・クラスタシステムを構成するサーバの台数を変更するクラスタ再構成技術に適合した、クライアントとサーバ・クラスタシステム間の負荷分散方法を提供する。
【解決手段】複数のクライアント100と、クライアント100からのリクエストを処理する複数のサーバ800を含み、複数のサーバの数を動的に変更するサーバ・クラスタ1100と、によって構成されるクライアント・サーバシステムに用いられる負荷分散方法であって、クライアント100は、サーバ・クラスタ1100を構成するサーバの数を検出し、サーバ数の増加が検出された直後は、該増加したサーバに対して送出されるリクエストの配分を他のサーバに比べて小さく設定し、前記設定された配分に基づいて前記複数のサーバに対してリクエストを送出する。
【選択図】　図１

特願２００３－２９１２８６

出　願　人　履　歴　情　報

識別番号　［０００００５１０８］

１．変更年月日　１９９０年　８月３１日
［変更理由］　新規登録
住　所　東京都千代田区神田駿河台４丁目６番地
氏　名　株式会社日立製作所